SOME COMPUTATIONAL ASPECTS IN VECTOR-VALUED PERFORMANCE INDEX OPTIMIZATION

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

MANOJ KUMAR DHOORIA

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JANUARY, 1987

CERTIFICATE

It is certified that this work entitled 'SOME COMPUTATIONAL ASPECTS IN VECTOR-VALUED PERFORMANCE-INDEX OPTIMIZATION' by Manoj Kumar Dhooria has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.

B. Sarkar

(B. Sarkar)
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur.

ACKNOWLEDGEMENT

I wish to express my sincere thanks to my thesis supervisor, Dr. Basanta Sarkar, for his wise counsel and encouragement at various stages of this thesis work. The literature provided by him and the useful discussions I had with him has helped immensely in the completion of this thesis.

Jan. 1987

Manoj Kumar Dhooria

ABSTRACT

Ever since the use of a vector performance-index in place of a scalar index in optimal control system problems was proposed by Zadeh in 1963, the idea has been under investigation.

Reid and Citron in 1971 proposed a good technique in which the vector optimization is done by optimizing a scalar auxiliary index equal to the sum of the components of vector index weighted by constant coefficients; it has been pointed out by many authors that it is very difficult to choose weighting coefficients for a complex system. To overcome this difficulty, Salukvadze in the same year (1971) proposed an alternative technique in which a norm approximation to the utopian point, in the performance-index space, defined by the optimum values of various component indices is optimized; but the techniques for the optimization of a general norm approximation were not available until recently. Nandakumar in 1986 proposed a numerical algorithm to optimize a general norm approximation function by combining the concepts of Reid and Citron and Salukvadze. But Nandakumar did not study the various properties, particularly, convergence-related properties, of his algorithm which are so important for computer implementation of his algorithm.

This thesis is a study of the various properties of Nandakumar's algorithm from the point of view of computer implementation of his algorithm. To ensure convergence of the algorit a real sequence is proposed the convergence properties of which are similar to those of Nandakumar's algorithm. In the process, a computer program has been developed which implements Nandakumar's algorithm for a class of systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND NOTATIONS

(Note: Only too frequently used symbols whose definition is not given at every place in the thesis are given here.)

$\underline{v}$	vector of those system variables with respect to which the vector-optimization is to be carried out
$\underline{q}$	parameter vector
$\underline{I}$	vector performance-index
$\emptyset$	admissible region of performance-index space
m	dimension of $\underline{v}$ or $\underline{q}$
n	dimension of $\underline{I}$
i	an integer that indicates a component of a vector
j	an integer that indicates an iteration of Nandakumar's algorithm
$\underline{c}$	n-dimensional vector of weighting constants
$\underline{I}^{o}$	utopian point in performance-index space
p	integer parameter that defines a general norm
s	step-size in Nandakumar's algorithm
a^{j}	a term of the real sequence that ensures convergence of Nandakumar's algorithm
$\underline{c}'$	value of $\underline{c}$ calculated at the end of an iteration of Nandakumar's algorithm
Q	region of parameter space where $\underline{q}$ is constrained to lie.

CHAPTER 1

INTRODUCTION

1.1 The Background

Conventionally, control systems have been optimized using a single number calculable from the system characteristics, called the scalar-valued performance-index, as a measure of the system performance. Problems arising out of the use of a single number as the performance-index have been pointed out ever since its inception (for control system problems) during World War II, but lacking a better alternative, it has been in widespread use.

Use of a vector-valued performance-index as an alternative to the scalar-valued performance-index in the problems of optimal control was suggested by Zadeh [1] in 1963. Since the vector-valued performance-index optimization as a design approach is still an unfamiliar field to most people, Appendix A has been added to this thesis as a brief review of this field and to introduce much used terminology of the field.

1.2 The Problem

A very general vector-valued performance-index optimization problem is now formulated for the sake of immediate discussion to follow.

For some system S, the vector-valued performance-index

$$\underline{I}(\underline{v}) = \begin{bmatrix} I_1(\underline{v}) \\ I_2(\underline{v}) \\ \vdots \\ I_n(\underline{v}) \end{bmatrix}$$

is considered, where $\underline{v}$ is m-dimensional vector of system variables (i.e., $\underline{v}$ may be state-vector $\underline{x}(t)$, control vector $\underline{u}(t)$, or parameter vector $\underline{q}$). Also, there may be some constraints imposed on $\underline{v}$.

The problem is to find out the "optimum" value of $\underline{I}(\underline{v})$, and the corresponding $\underline{v}$.

1.2.1 Remark 1

Throughout the rest of this thesis, the word "optimum" is used as synonymous with "minimum". Case of maximization is exactly similar except that the sign of the function to be optimized is to be reversed.

1.2.2 Remark 2

In what follows, the statements "system S is optimum" and "performance-index $\underline{I}$ is optimum" are treated as equivalent.

1.3 Work of Reid and Citron [2]

This work gives a technique to solve the problem defined in Section 1.2.

Since an n-dimensional performance-index $\underline{I}$ is being considered and each component of $\underline{I}$ is a real number, $\underline{I}$ can be represented in an n-dimensional vector space (performance-index space) defined over the field of real numbers $\mathbb{R}$. It is assumed that all performance-indices $\underline{I}$ can be represented in the first orthant $\mathbb{R}^{n+}$ of this n-dimensional performance-index space (i.e., each component of $\underline{I}$ is positive); this can be ensured by changing the sign of those scalar performance-indice which have a negative sign. $\mathbb{R}^{n+}$ consists of two mutually exclusive regions:

1. Admissible region $\emptyset$. This corresponds to the performance-indices obtained by those system variables $\underline{v}$ which satisfy the various constraints for the specific problem.
2. Inadmissible region Ω.

Obviously, the search for non-inferior index vectors should be confined to $\emptyset$. Also, origin can be safely excluded from $\emptyset$ because this is an "optimal" point in $\mathbb{R}^{n+}$ and is a trivial case (and often unrealizable case, because the minimum value of an individual performance-index will normally be greater than zero). And the discussion of Reid and Citron does not apply to the case when origin is an "optimal" (also noninferior) point.

On p. 16, Reid and Citron prove that <u>for non-inferior vectors to exist</u>, following two conditions should hold.

1. Region $\emptyset$ should be at least partially closed, and noninferior solutions can exist only over this closed boundary of region $\emptyset$.

2. Noninferior solutions can exist only on the lower boundary of region $\emptyset$.

Figure 1.1, curve (I) shows this for 2-dimensional case when whole of $\emptyset$ is closed.

Now a method to determine the lower boundary of admissible region $\emptyset$, i.e., noninferior solutions, can be considered.

Converting the noninferior index vector problem into a family of scalar index problems appears to be a good technique since the later type of problems are well studied by modern control theorists. Therefore, the auxiliary scalar index J is considered which is a function of the index vector $\underline{I}$. The index J will normally have one minimum value which will correspond to one point on the lower boundary of $\emptyset$. So, to specify the whole of lower boundary, an n-dimensional vector parameter $\underline{c}$ is introduced and the scalar performance-index is defined as

$$J \triangleq F(\underline{I}, \underline{c}) \qquad (1.1)$$

which can be minimized for different values of $\underline{c}$ to obtain different noninferior solutions.

What should be the form of $F(\underline{I}, \underline{c})$? The desired construction of $F(\underline{I}, \underline{c})$ would be such that minimum value of J corresponds to the lower boundary points of $\emptyset$.

"In general, it is difficult to find a functional form for . . ." J ". . . that will ensure a one-to-one correspondence between minimum points of . . ." J ". . . for a given value of . . ." $\underline{c}$ ". . . and noninferior points on . . ." the lower

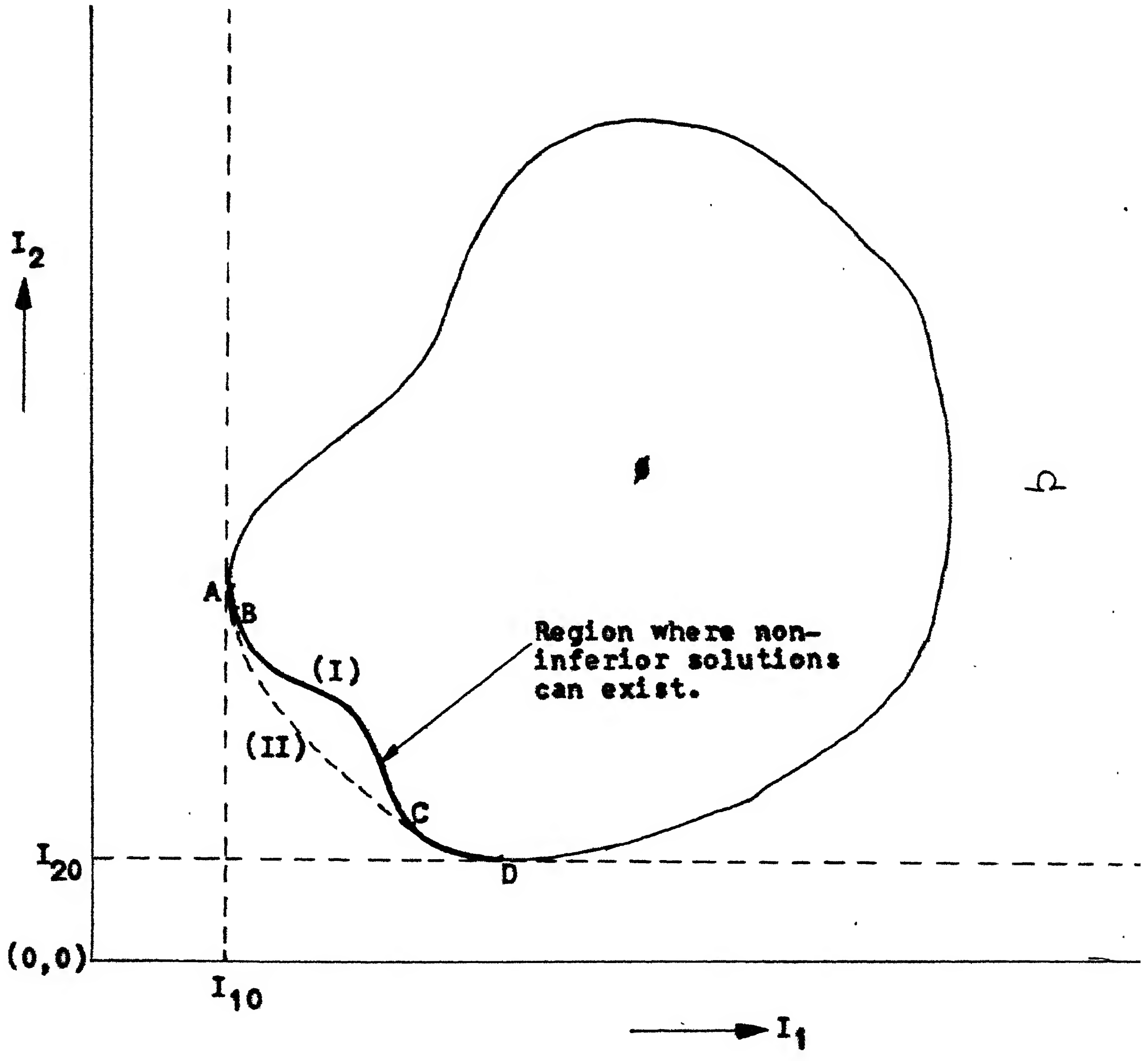

Fig. 1.1: Illustration of that region of performance-index spac
where noninferior solutions can exist.

boundary of $\emptyset$. "Therefore, consider now a limited class of problems . . ." for which this correspondence holds true. "This form is the standard linear sum of scalar indices weighted by constant coefficients given as . . ." [2 , pp. 16-19].

$$J = \langle \underline{c}, \underline{I} \rangle ; \quad c_i > 0; \quad i = 1, 2, \ldots, n \qquad (1.2)$$

Then Reid and Citron give a geometrical proof that Eq. (1.2) indeed gives the lower boundary of $\emptyset$ by minimizing J for various values of $\underline{c}$ so long as the shape of lower boundary is such that 1:1 correspondence between minimum points of J and values of $\underline{c}$ hold. Implicit in this geometric proof is the assertion that 1:1 correspondence between values of $\underline{c}$ and minimum points of $J = \langle \underline{c}, \underline{I} \rangle$ holds true if following two conditions are satisfied.

1. Noninferior index surface is monotonically varying with respect to each scalar index I_i.
2. If the equation of noninferior index surface is given in terms of various I_i, $i = 1, 2, \ldots, n$, then first partial derivatives $\partial I_n / \partial I_i$, $i = 1, 2, \ldots, n-1$, are also monotonically varying.

Implication of this is that the results of Reid and Citron cannot give all the noninferior solutions when noninferior index surface is like the lower boundary of Figure 1.1, curve (I), but can give all the solutions when the lower boundary is like the noninferior index surface of Figure 1.1, curve (II).

In the case of Figure 1.1, curve (I), only that portion of noninferior index surface will be obtained by this method where this correspondence holds true, i.e. parts AB and CD.

1.4 Work of Salukvadze [3, 4]

To use Reid and Citron's [2] technique, $\underline{c}$ is to be chosen before the solution is attempted. This is an especially difficult task for large systems, as noted by Salukvadze [3] and Nandakumar[5]. To overcome this difficulty, Salukvadze gives another technique to solve vector optimization problem.

Let each component of $\underline{I}$ be minimized separately. Let the minimum value of I_i be I_{io}, $i = 1, 2, \ldots, n$. Ideally, one wants $\underline{I} = [I_{10}\ I_{20}\ \ldots\ I_{n0}]^T$ as the solution, but this is not possible in majority of practical cases since, in most cases, same $\underline{v}$ will not minimize more than one component of $\underline{I}$. So, the point

$$\underline{I}^o = \begin{bmatrix} I_{10} \\ I_{20} \\ \vdots \\ I_{n0} \end{bmatrix} \qquad (1.3)$$

in performance-index space is the ideal point one is striving for, but cannot get it because of the very choice of various mutually conflicting criteria in vector performance-index $\underline{I}$; point $\underline{I}^o$ is called (by Salukvadze) the "utopian point".

Since the utopian point cannot be reached, one can think of going closest to it. So an approximation function (approximation to point $\underline{I}^o$) $R(\underline{I}) = R(\underline{I}(\underline{v}))$ is introduced. Minimum value of $R(\underline{I})$ will give corresponding $\underline{v}$ as the "optimum" solution in $R(\underline{I})$-sense. Moreover, approximation function $R(\underline{I})$ may be <u>any</u> scalar positive definite function of $\underline{I}$. <u>Once a specific $R(\underline{I})$ has been selected, the definition of the type of compromise one is looking for is fixed</u>.

As a useful approximation function $R(\underline{I})$, Salukvadze considers the Euclidian norm squared:

$$R(\underline{I}) = ||\underline{I} - \underline{I}^o||^2 = \sum_{i=1}^{n} (I_i - I_{io})^2 \tag{1.4}$$

when all components I_i of $\underline{I}$ are normalized, i.e., reduced to dimensionless form. If it is not so, then

$$R(\underline{I}) = \sum_{i=1}^{n} \left(\frac{I_i - I_{io}}{I_{io}}\right)^2 \tag{1.5}$$

can be considered.

Subsequently, more general norm functions as approximation functions were considered. According to Nandakumar [5], p. 6, following generalized norm was first introduced by Yu:

$$R(\underline{I}) = \left[\sum_{i=1}^{n} (I_i - I_{io})^p\right]^{1/p} \tag{1.6}$$

where p is a positive integer.

Its counterpart according to Eq. (1.5) when various components of $\underline{I}$ are not reduced to dimensionless form can easily be written as

$$R(\underline{I}) = \left[\sum_{i=1}^{n} \left(\frac{I_i - I_{io}}{I_{io}}\right)^p\right]^{1/p} \tag{1.7}$$

In Eqs. (1.6) and (1.7), selection of a specific p means the definition of compromise has been specified.

By minimizing any R out of Eqs. (1.4) to (1.7), a minimizing $\underline{v}$ is obtained. With the selection of this $\underline{v}$, ". . . some deterioration takes place in each quality indicator . . ." I_i ". . . of the system (compared to what we would have obtained by optimization of only that particular indicator); however, this deterioration is uniform with respect to the entire collection of indicators . . ." I_i ". . . and is minimal possible." [4 , pp. 1170-1171].

1.5 Work of Nandakumar[5]

Literature before the work of Nandakumar do not provide a suitable approach to solve the vector optimization problem using Eq. (1.6) for $p > 2$; even for $p = 2$, Salukvadze's [3] approach is quite involved. Nandakumar has given a numerical algorithm to solve the problem; he has also indicated a way to choose a specific compromise solution, i.e., a specific p, for a given system.

1.5.1 Minimization of the Performance-Index Given by Eq. (1.6) For a Given p

Performance-index given by Eq. (1.6) is rewritten as follows:

$$J = \left[\sum_{i=1}^{n} (I_i - I_{io})^p \right]^{1/p} ; \quad p = 1, 2, \ldots, \infty \qquad (1.8)$$

where I_{io} is the minimum value of I_i.

This equation gives a noninferior solution, for a given p, on the lower boundary of the admissible region $\emptyset$ of performance-index space [3].

Another way of finding the lower boundary of the admissible region $\emptyset$ of performance-index space is to minimize

$$J = \langle \underline{c}, \underline{I} \rangle = \sum_{i=1}^{n} c_i I_i ; \quad c_i > 0; \quad i = 1, 2, \ldots, n \qquad (1.9)$$

Eq. (1.9) is easy to minimize but it is very difficult to choose $\underline{c}$ for a practical system. Eq. (1.8), on the other hand, is difficult to minimize but is free from the problems associated with choosing $\underline{c}$. Nandakumar has given an iterative algorithm to minimize Eq. (1.8) by reducing Eq. (1.8) to take the form of Eq. (1.9).

Minimization of Eq. (1.8) is equivalent to minimizing

$$J = \sum_{i=1}^{n} (I_i - I_{io})^p \qquad (1.10)$$

In Eq. (1.10), put

$$c_i = (I_i - I_{io})^{p-1} ; \quad i = 1, 2, \ldots, n \qquad (1.11)$$

So, Eq. (1.10) can be written as

$$J = \sum_{i=1}^{n} c_i (I_i - I_{io}) = \sum_{i=1}^{n} c_i I_i - \sum_{i=1}^{n} c_i I_{io} \qquad (1.12)$$

Since I_{io} is the minimum value of I_i, c_i given by Eq. (1.11) is always positive. So, minimization of Eq. (1.10) is equivalent to minimization of Eq. (1.9) subject to Eq. (1.11), because if c_i is treated as constant, then $\sum_{i=1}^{n} c_i I_{io}$ in Eq. (1.12) becomes constant.

So, <u>the minimization of Eq. (1.8), which is the goal of Nandakumar's thesis, is equivalent to minimization of Eq. (1.9) subject to c_i defined by Eq. (1.11)</u>. A way of achieving this is an iterative algorithm in which one starts with an assumed $\underline{c}$ and minimizes Eq. (1.9) <u>independent of</u> Eq. (1.11); $\underline{I}$ obtained by this minimization is used to calculate new $\underline{c}$ using Eq. (1.11). Then one checks if assumed $\underline{c}$ is equal to calculated $\underline{c} = \underline{c}'$; if yes, then the problem is solved; if no, then modify the assumed $\underline{c}$ for next iteration by equation

$$c_i = c_i + s \cdot (c_i' - c_i); \quad i = 1, 2, \ldots, n \qquad (1.13)$$

where s is a suitably chosen step-size.

And this is repeated till convergence.

Nandakumar also says (without giving any reasons) on p. 16 of his thesis that it ". . . is convenient to normalize the weighting constants . . ." c_i such that

$$\sum_{i=1}^{n} c_i = 1; \; c_i > 0 \qquad (1.14)$$

which means the iterative algorithm, instead of calculating $\underline{c}$ from Eq. (1.11), should use the following equation for the purpose:

$$c_i = \frac{(I_i - I_{io})^{p-1}}{\sum_{j=1}^{n} (I_j - I_{jo})^{p-1}}; \quad i = 1,2,\ldots,n \qquad (1.15)$$

This method will work (provided the solution exists), theoretically, for all finite p.

Nandakumar has given the above procedure to calculate solution for finite p in the form of a step-by-step procedure. In Figure 1.2, Nandakumar's algorithm is written as a flow-chart which is exactly equivalent to his written statements, except one difference; while Nandakumar has indicated the system variables (control vector $\underline{u}$, or parameters) with respect to which the minimization is to be carried out, these have not been indicated in Figure 1.2; this does not make any difference if it is kept in mind that minimization can only be carried out with respect to certain system variables, $\underline{v}$ in the context of this thesis.

Nandakumar has shown on p. 13 of his thesis that for $p = \infty$, the solution of Eq. (1.8) has the property that

$$I_i - I_{io} = \text{constant}; \quad i = 1, 2, \ldots, n \qquad (1.16)$$

1.5.2 Selection of a Specific Compromise Solution

Now that a set of compromise solutions corresponding to different values of p have been obtained, a specific solution is to be selected.

Many authors have suggested various criteria to select a specific solution [1, 2], none of which appears to be of universal application. Nandakumar offers yet another approach to do this job.

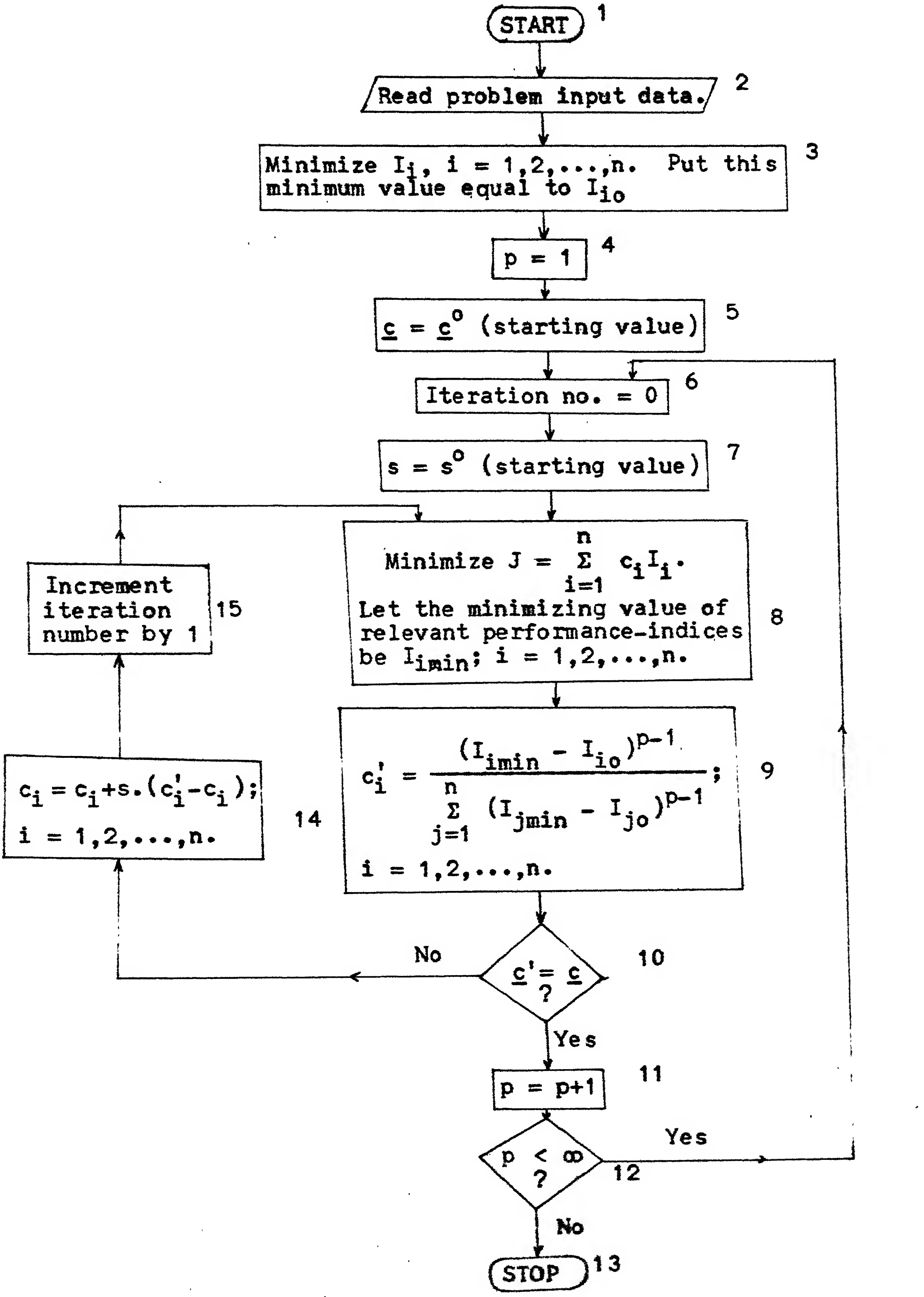
START 1
Read problem input data. 2
Minimize I_i, $i = 1,2,\dots,n$. Put this minimum value equal to I_{io} 3
$p = 1$ 4
$\underline{c} = \underline{c}^o$ (starting value) 5
Iteration no. = 0 6
$s = s^o$ (starting value) 7
Minimize $J = \sum_{i=1}^{n} c_i I_i$. Let the minimizing value of relevant performance-indices be I_{imin}; $i = 1,2,\dots,n$. 8
$c_i' = \frac{(I_{imin} - I_{io})^{p-1}}{\sum_{j=1}^{n} (I_{jmin} - I_{jo})^{p-1}}$; $i = 1,2,\dots,n$. 9
$\underline{c}' = \underline{c}$? 10
No
Yes
$p = p+1$ 11
$p < \infty$? 12
Yes
No
STOP 13
$c_i = c_i + s.(c_i' - c_i)$; $i = 1,2,\dots,n$. 14
Increment iteration number by 1 15

Fig. 1.2: Flow-chart equivalent of Nandakumar's algorithm for

Since every p is associated with a specific compromise solution $\underline{I} = [I_1 \ \ I_2 \ \dots \ I_n]^T$, based on specific system requirements the designer can fix the value of one of the n scalar indices; Nandakumar calls this the "reference index". This fixes p and, consequently, the solution vector $\underline{I}$. So, the job is done.

1.6 Objective of This Thesis

Nandakumar's [5] thesis gives an algorithm to compute the various compromise solutions of a general vector-optimization problem and a method to choose one compromise solution out of the set of compromise solutions computed by the algorithm. Simplicity of this algorithm to tackle difficult problems is its beauty.

But Nandakumar did not attempt a comprehensive study of the various properties of his algorithm such as, when the algorithm converges to a solution, on which parameters of his algorithm the convergence property depends, how the rate of convergence can be accelerated, etc.

The aim of this thesis is to study the properties of Nandakumar's algorithm and to develop a computer program which implements Nandakumar's algorithm for a class of vector-optimization problems. Out of the various properties studied, convergence property is given special emphasis because of its extreme importance in computer implementation of the algorithm. A real sequence whose convergence properties are similar to the convergence properties of Nandakumar's algorithm has been proposed to ensure convergence of Nandakumar's algorithm.

1.7 Organization of This Thesis

Chapter 2 is a theoretical probe into the various properties of Nandakumar's algorithm from the point of view of computer implementation. Chapter 3 discusses the computational problems and how the study of Chapter 2 has been used in the computer program developed. Chapter 4 is the concluding chapter which suggests a few areas of further work on the subject.

CHAPTER 2

ANALYSIS OF NANDAKUMAR'S ALGORITHM

2.1 Introduction

Before saying anything about the implementation problems, it will be good to state the assumptions about the class of problems to which Nandakumar's algorithm is applicable; these were not stated by Nandakumar but they are important for the results of Reid and Citron [2] to be valid and were assumed by Reid and Citron, as discussed in Section 1.3.

1. It is assumed that $I_i(\underline{v})$, $i = 1, 2, \ldots, n$, are positive.
2. Noninferior index surface is monotonically varying with respect to each I_i, $i = 1, 2, \ldots, n$.
3. If the equation of noninferior index surface is given as a function of the various I_i, $i = 1, 2, \ldots, n$, then the first partial derivatives $\partial I_n / \partial I_i$, $i = 1, 2, \ldots, n$, are also monotonically varying.

There is no simple way of ensuring that assumptions 2 and 3 hold for a specific problem, as implied by Reid and Citron [2]; probably some heuristic considerations can help.

Also, implicit in the use of Eq. (1.6) by Nandakumar to find noninferior solutions is another assumption, as indicated by Salukvadze [3, 4] but not mentioned by Nandakumar.

4. All components of $\underline{I}$ are reduced to dimensionless form.

If this is not so, then Eq. (1.7) instead of Eq. (1.6) should be used.

Now follows an analysis of the reasons why $\underline{c}$ was normalized by Nandakumar in the sense that

$$\sum_{i=1}^{n} c_i = 1, \tag{2.1}$$

and of the conditions that can ensure the convergence of Nandakumar's algorithm to a solution.

2.2 Normalization of $\underline{c}$

Nandakumar [5] has mentioned on p. 16 of his thesis that it is ". . . convenient to normalize the weighting constants . . . c_i in the sense of Eq. (2.1). The reason this normalization approach is convenient can be seen from the following discussion.

If assumed c_i in the beginning of an iteration of Nandakumar's algorithm and $c_i' = (I_i - I_{io})^p$ calculated at the end of this iteration differ by orders of magnitude, some iterations are needed to make the orders of magnitude of c_i and c_i' conform. Only then the convergence point where

$$\underline{c} = \underline{c}' \tag{2.2}$$

can be approached. This is because of the following reasons.

Since c_i, i = 1, 2,...,n are all positive, one can write

$$J = \sum_{i=1}^{n} c_i I_i = (\sum_{i=1}^{n} c_i) \cdot [\sum_{i=1}^{n} (\frac{c_i}{\sum_{j=1}^{n} c_j} \cdot I_i)] \tag{2.3}$$

Solution $\underline{I}$ which minimizes Eq. (2.3) is the same as that which minimizes

$$J = \sum_{i=1}^{n} \left(\frac{c_i}{\sum_{j=1}^{n} c_j} \cdot I_i \right)$$

because $\sum_{i=1}^{n} c_i$ is a constant. This means, $\underline{I}$ which minimizes $\sum_{i=1}^{n} c_i . I_i$ depends on the relative values of c_i, $i = 1, 2, \ldots, n$. Experience of various tests on computer has shown that the process of making the orders of magnitude of c_i and c_i' conform does little about adjusting the relative values of various components of $\underline{c}$ to that of final solution; these iterations of Nandakumar's algorithm are almost wasted. In some tests with a simple second order linear system with 2-dimensional $\underline{I}$, CPU time thus wasted was found to be approximately equal to the time it takes to solve the problem with normalized $\underline{c}$.

Normalized $\underline{c}$ was also found useful in developing convergence criteria for Nandakumar's algorithm, as will become apparent in next two sections.

2.3 Definition of the Point of Convergence

Nandakumar's [5] definition of the point where we say that the algorithm has converged, Eq. (2.2), is to be modified to suit computer implementation needs because nothing is exact on computer.

For a precise mathematical definition suitable for computer implementation, a clue is suggested by Kreyszig [6], pp. 650-651, during a discussion of infinite sequences. Rele- [illegible]ion is quoted below.

"If to each positive number n there is assigned a number z_n, then these numbers

$$z_1, z_2, \ldots, z_n$$

are said to form an infinite sequence or, briefly, a sequence, and the numbers z_n are called the terms of the sequence.

A sequence whose terms are real is called a real sequence."

"A sequence $z_1, z_2, \ldots$ is said to converge . . . if there is a number c with the following property. For every positive real ε (no matter how small but not zero) we can find an integer N such that

$$|z_n - c| < \varepsilon \qquad \text{for all} \quad n > N.$$

c is called the limit of the sequence."

This suggests that if an appropriate real number a^j could be associated with $j^{\underline{th}}$ iteration of Nandakumar's algorithm such that the convergence properties of the real sequence $a^1, a^2, \ldots, a^j, \ldots$ are similar to the convergence properties of Nandakumar's algorithm, then the definition of the point of convergence of Nandakumar's algorithm could have been borrowed from Kreyszig [6].

The ultimate aim of making $\underline{c} = \underline{c}'$ suggests two ways to construct such a sequence:

1. $$a^j \triangleq ||\underline{c} - c'|| \; ; \; j = 1, 2, \ldots$$

   If algorithm converges, limit of this sequence should be zero.

For computer implementation, Nandakumar's algorithm can be considered to have converged when

$$a^j = ||\underline{c} - \underline{c}'|| \leq \varepsilon$$

2. A set of n sequences, which are to converge simultaneously, is formed by defining

$$a_i^j \triangleq |c_i - c_i'|; \quad i = 1, 2, \ldots, n; \; j = 1, 2, \ldots \qquad (2.4)$$

If algorithm converges, the limit of each such sequence shall be zero.

For computer implementation, Nandakumar's algorithm can be considered to have converged when

$$a_i^j = |c_i - c_i'| \leq \varepsilon_i; \quad i = 1, 2, \ldots, n \qquad (2.5)$$

This second definition will be used in this thesis.

2.4 Conditions for Convergence

In Nandakumar's algorithm, if the point of convergence is not reached, we need modify $\underline{c}$ for next iteration by

$$c_i = c_i + s \cdot (c_i' - c_i); \quad i = 1, 2, \ldots, n \qquad (2.6)$$

and go for next iteration.

Since the algorithm is general, possibility of knowing the analytical relationships between $\underline{v}$ and the various performance-indices is ruled out. So an analysis of Nandakumar's algorithm to know whether the algorithm converges before it is actually used cannot be done; only numerical methods are feasible.

The numerically feasible test that was found suitable for implementation is stated as a theorem by Kreyszig [6], p. 661, in the following words:

"If a real sequence is bounded and monotone, it converges."

That the sequence a_i^1, a_i^2,... is bounded (see Footnote 1) can be seen from the definition of a_i^j, Eq. (2.4), because both c_i and $c_i^!$ are between zero and one so that a_i^j is finite.

Since the limit of the sequence a_i^1, a_i^2,... is zero and, in general, the algorithm is started with a finite a_i^1, it is to be ensured that the sequence is monotonically decreasing, i.e.,

$$a_i^1 \geq a_i^2 \geq \ldots$$

To ensure that the sequence a_i^1, a_i^2,... is monotonically decreasing, it is <u>assumed</u> that c_i versus $c_i^!$ curve is monotone, as shown in Figure 2.1, curves (I) and (II) (see Footnote 2). That now monotonicity of the sequence a_i^1, a_i^2,... is assured can be explained as follows.

In curve (I) of Figure 2.1, points A and B mark the extreme values of c_i and $c_i^!$, and point C is the actual solution.

---

1 "A sequence z_1, z_2,... is said to be <u>bounded</u>, if there is a positive number K such that . . .

$|z_n| < K$ for all n." - Kreyszig [6], p. 653.

2 In all examples given by Nandakumar [5] except one (Example 6.2), and in both examples given in this thesis, curve (I) holds. In one example in Nandakumar's thesis for which <u>I</u> was 3-dimensional, curve (I) holds for two values of i and curve (II) holds for one i. All of the testings for this thesis were confined to curve (I).

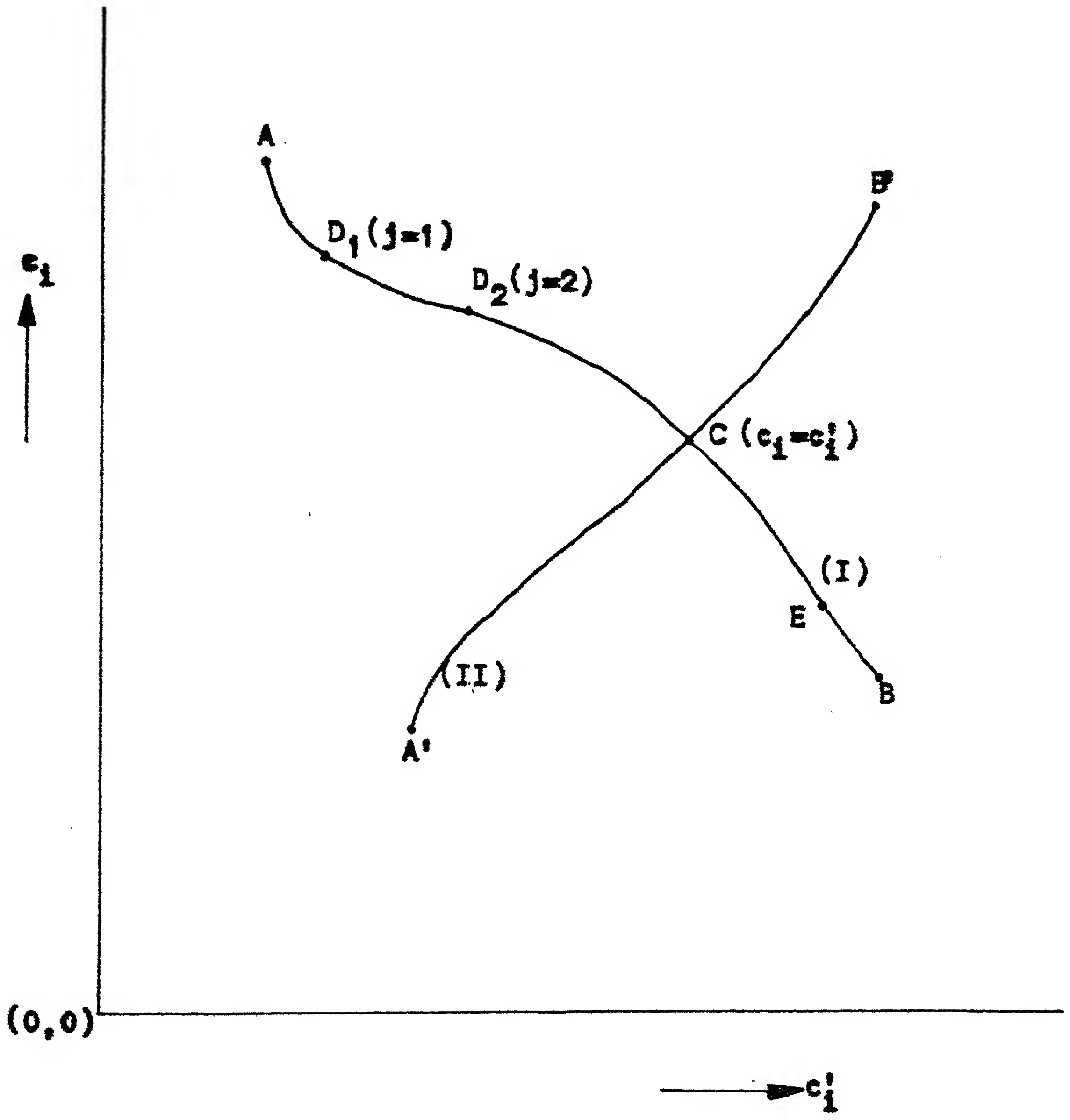

Fig. 2.1: c_i versus c_i' curve should be monotone: Illustration

It is assumed that the iterations are started at point D_1 where $c_i > c_i'$.

Algorithm to generate c_i for next iteration can be designed to ensure that the relations

$$a_i^1 \geq a_i^2 \geq \ldots$$

hold.

If Eq. (2.4) is used to form the sequence, then monotonically decreasing sequence may involve oscillation of points D_j, $j = 2, 3, \ldots$ about solution point C. It appears intuitively more appealing if D_j could be made to steadily approach point C, without oscillating about the solution point. This can be assured if the definition of Eq. (2.4) is modified as follows:

$$a_i^j = c_i - c_i'; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots \qquad (2.7)$$

Ways of ensuring the assumption of Figure 2.1 before Nandakumar's algorithm is run are not known, but the practical feasibility of this assumption was brought out by the fact that for all examples for which Nandakumar's algorithm was run, this assumption was valid (see Footnote 2).

So, the conclusions are:

1. It is to be ensured that the magnitude of a_i^j calculated by Eq. (2.7) monotonically decreases in successive iterations and that the sign of a_i^j does not change in successive iterations. (Since the limit of a_i^j is zero, a change in its sign means the point of convergence has been bypassed).

This will ensure that Nandakumar's algorithm is steadily converging to a solution. How this will be ensured will be discussed in Chapter 3.

2. It is to be considered that the algorithm has converged when conditions dictated by Eq. (2.5) are satisfied.

CHAPTER 3

IMPLEMENTATION OF NANDAKUMAR'S ALGORITHM

3.1 Outline

Section 3.2 discusses the computational difficulties from the point of view of Nandakumar's general algorithm. Section 3.3 gives the details of the FORTRAN program developed to implement Nandakumar's algorithm. Examples are given in Section 3.4.

3.2 On General Algorithm

Discussions in this section have reference to Nandakumar's algorithm as given in Figure 1.2.

The minimum information needed to run the algorithm are

1. m, the dimension of $\underline{v}$.
2. n, the dimension of $\underline{I}$.
3. p. Nandakumar has written the algorithm in a form in which it can be run for all finite 'p' in a single run. Discussion to follow removes this outer p-loop; this does not make any difference to computational problems of the algorithm.
4. vector performance-index $\underline{I}(\underline{v})$. This information is supplied by providing a subroutine that will return $I_i(\underline{v})$ once $\underline{v}$ and i are given.
5. constraints on $\underline{v}$. In an implementation scheme, this information is supplied by providing a subroutine which can check if a given $\underline{v}$ satisfies the constraints.

With this information, it is possible to minimize $I_i(\underline{v})$, $i = 1, 2, \ldots, n$ as well as $J(\underline{v}) = \sum_{i=1}^{n} c_i . I_i(\underline{v})$.

Now the computational problems of Nandakumar's algorithm are taken up on block-by-block basis as per Figure 1.2.

3.2.1 Block 5

Algorithm can only be started with an initial value of $\underline{c}$. Keeping in mind the normalization constraint imposed by Eq. (2.1) on $\underline{c}$, and a complete ignorance as to what $\underline{c}$ should be to solve the problem at hand, it appears most obvious to choose (and that this choice is good enough was brought out by various tests on computer) initial $\underline{c}$ such that all of its components are equal, i.e.,

$$c_i = \frac{1}{n} \tag{3.1}$$

because $\underline{c}$ is n-dimensional.

3.2.2 Blocks 6 and 15

In Figure 1.2, iteration number is not being used for any purpose. Well, in this type of iterative algorithm, solution shall be obtained, in general, after infinite iterations. Since this is impractical, an upper limit is to be imposed on the number of iterations to be executed in the worst case. Within a given number of iterations one should get the solution with reasonable accuracy. If this does not happen, one of the many possible actions can be to restart the algorithm with the value of $\underline{c}$ obtained at the end of the given number of iterations if the algorithm is found converging till this point. In this thesis, if maximum allowable number of iterations are exceeded, algorithm is declared "not converging" and appropriate final data ($\underline{c}$, $\underline{c}'$, $\underline{I}$, etc.) are printed out.

3.2.3 Block 10

Convergence of the algorithm to a solution for a given p is tested here, as discussed in Section 2.3, by Eq. (2.5) but in a bit modified form as follows:

$$\frac{|c_i - c_i'|}{c_i} \leq \alpha ; \quad i = 1, 2, \ldots, n, \qquad (3.2)$$

where α is a real positive number less than one.

Need for this form was felt during actual implementation. $\alpha = 0.02$ was found useful and has been implemented.

Since both $\underline{c}$ and $\underline{c}'$ are normalized, their comparison can be made by testing only (n-1) corresponding components instead of testing all the components; but in this thesis, all the n components are compared because the CPU time saved on this count is very small.

3.2.4 Blocks 7 and 14

If, for a specific iteration j, the algorithm does not pass the test of block 10, $\underline{c}$ is to be modified for next iteration in block 14; and what this modified $\underline{c}$ will be depends upon the step-size 's'. Whether, for a specific iteration, the algorithm passes the test of block 10 depends on $\underline{c}$ chosen for this iteration. So, step-size 's' is to be carefully chosen.

According to the discussion of Section 2.4, Nandakumar's algorithm will be steadily converging to a solution provided that the magnitude of a_i^j calculated by Eq. (2.7) is monotonically decreasing in successive iterations and that the sign of a_i^j does not change in successive iterations. So, the algorithm for step-size s should ensure that the sign of a_i^j does not change in successive iterations and the absolute value of a_i^j does not increase in successive iterations. An iteration for which a_i^j, $j \geq 2$, does not violate these two conditions is termed "successful", else it is declared "unsuccessful"; in either case, step-size s should be changed, directly or indirectly (see Footnote 3). In the implementation for this thesis, block 7 which initializes 's' was removed and a block just before block 14 was added to generate 's' by the algorithm on next page.

3 Indirectly changing 's' means c for next iteration is changed by some means other than changing 's' explicitly.

ALGORITHM TO GENERATE STEP-SIZE 's'

```
START
  If iteration number j = 1, then
    declare the iteration "successful".
    initialize step-size s = 0.5.
    $a_i^j = c_i^j - c_i'^j$; j = 1, 2,...,n.
  else
    $a_i^j = c_i^j - c_i'^j$; j = 1, 2,...,n.
    If $b_i = \frac{a_i^{j-1}}{a_i^j} \geq 1$ for all j = 1, 2,...,n, then
      declare the iteration "successful".
      $s \leftarrow (s \times 1.2)$.
    else
      declare the iteration "unsuccessful".
RETURN
```

Once an iteration is declared "unsuccessful", the proper c certainly lies between c for this "unsuccessful" iteration and the previous "successful" iteration. Fastest rate of convergence was obtained by making the c for next iteration equal to the average of c for the most recent "successful" and "unsuccessful" iterations; direct change of 's' is not needed now. This has been implemented.

Also, it was found that sometimes, particularly for values of p $\geq$ 5, a c_i^j gives an $a_i^j = (c_i^j - c_i'^j) > 0$ and c_i^{j+1} for next iteration which differs from c_i^j by 0.0005 gives an $a_i^{j+1} < 0$. In such cases, c_i^j accurate to three decimal places was declared the final solution even though the conditions of Eq. (3.2) were not satisfied. The reason was traced down to the flatness of c_i versus c_i' curve; two approximate plots for a second order system with 2-dimensional $\underline{I}$ to be taken up as Example 2 in Section 3.4 are shown in Figure 3.1.

Also, an output block missing in Figure 1.2 is to be added.

3.3 Details of the Program Developed

A computer program has been developed which implements Nandakumar's algorithm. In view of the limited time available and extreme generality of Nandakumar's algorithm, the program developed is applicable only to a subset of the problems to which Nandakumar's general algorithm is applicable. Specifically, the problems for which this program is applicable are the following limited class of problems:

1. System S being considered is time-invarient.
2. Behaviour of the system can be completely characterized by a set of 'm' free parameters $\{q_1, q_2, \ldots, q_m\}$, each of which is a real number (see Footnote 4). This free

---

[4] Actually q_i will be, more often, a coefficient - i.e., a function of system parameters. But throughout this thesis, this will be called a parameter.

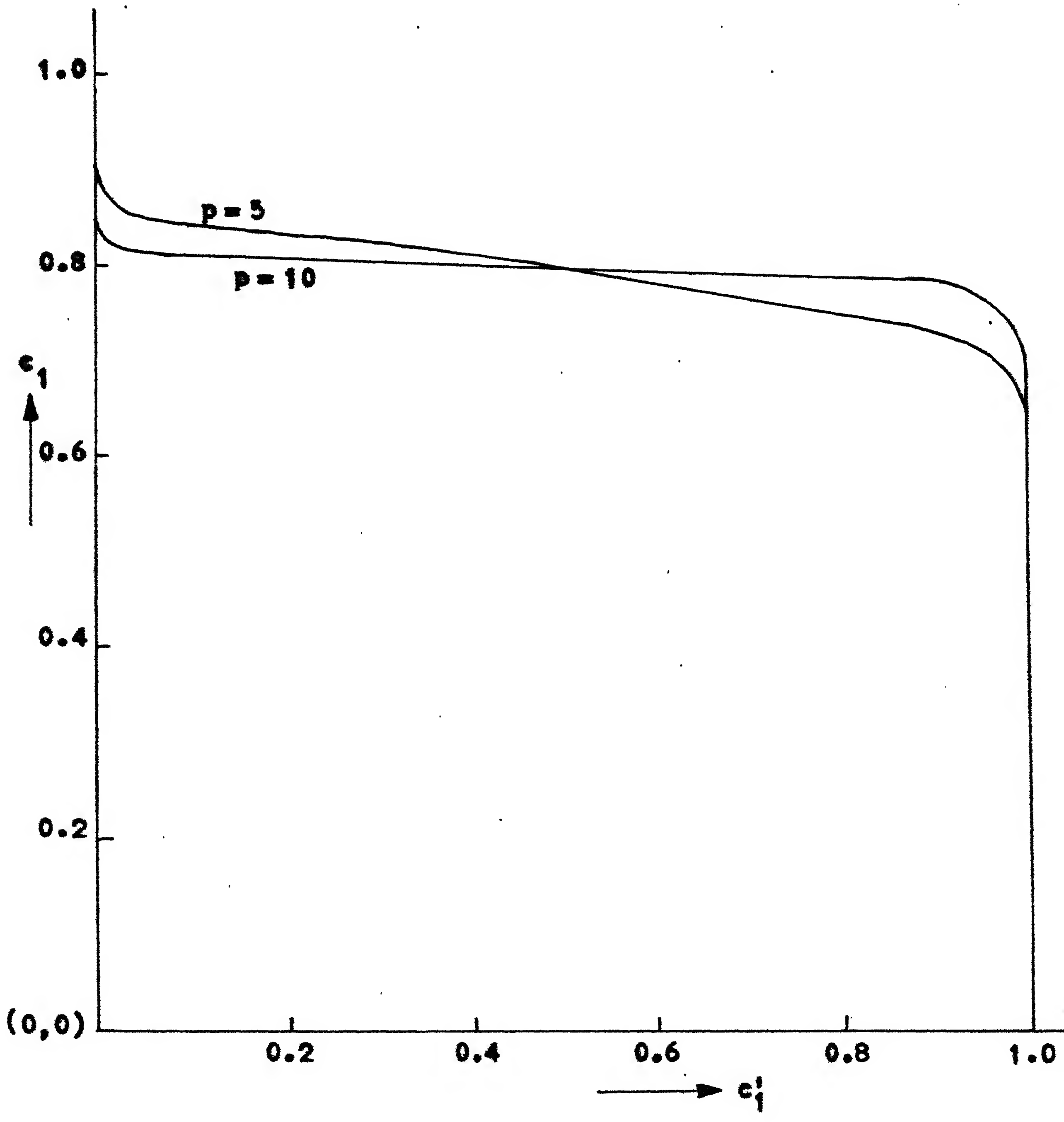

Fig. 3.1: c_i versus c_i' curve for Example 2, Section 3.4.

parameter set shall be written as the parameter vector

$$\underline{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix}$$

3. The parameter vector $\underline{q}$ defines a vector-space. Because of physical limitations, $\underline{q}$ cannot assume an arbitrary value. This thesis assumes that $\underline{q}$ is constrained to lie in region Q of parameter-space (see Footnote 5).

All testings for this thesis have been done for this limited class of systems.

Actual implementation required a subroutine to minimize a scalar function of n-variables subject to general constraints. This subroutine named MINIM in the program of this thesis uses Pattern-Search Algorithm (Appendix B) which is simple, sufficiently effective and easy to implement on computer.

Nandakumar's algorithm yields a compromise solution in the sense that it tries to minimize

$$J(\underline{q}) = [\sum_{i=1}^{n} (I_i(\underline{q}) - I_{io})^p]^{1/p} ,$$

[5] This region Q may be characterized by a set of equality and inequality constraints on the elements of $\underline{q}$ or functions thereof, or may be characterized otherwise.

the solution vector $\underline{I}$ of which is an approximation to the utopian point $\underline{I}^o$. But as discussed by Salukvadze [3, 4], this could have been just any positive definite approximation to $\underline{I}^o$. Program of this thesis has been written in such a way that c_i can be any arbitrary function for which the solution exists. So $J(\underline{q})$ can assume any form which can be reduced to

$$J(\underline{q}) = \sum_{i=1}^{n} c_i \cdot I_i(\underline{q}).$$

All that is to be done for this is to replace that routine which calculates $\underline{c}$ by appropriate routine; this subroutine should, however, ensure that $\sum_{i=1}^{n} c_i = 1$.

Also, function minimization routine MINIM requires a starting $\underline{q} = \underline{q}^o$. To start Nandakumar's algorithm's iterative loop, a suitable $\underline{q}^o$ was found to be the average of various $\underline{q}^i$ which minimize different performance-indices individually in block 3, Figure 1.2.

Keeping in mind the discussions of this thesis, flow-chart of Figure 1.2 is modified in Figure 3.2. Program of this thesis is an implementation of the flow-chart of Figure 3.2.

3.4 Examples

Using the program developed, two examples have been worked out. Each of these examples uses a 2-dimensional $\underline{I}$. The first example was solved by Nandakumar, so results can be compared. The second example uses two well studied criteria for parameter optimization as the elements of the vector criterion.

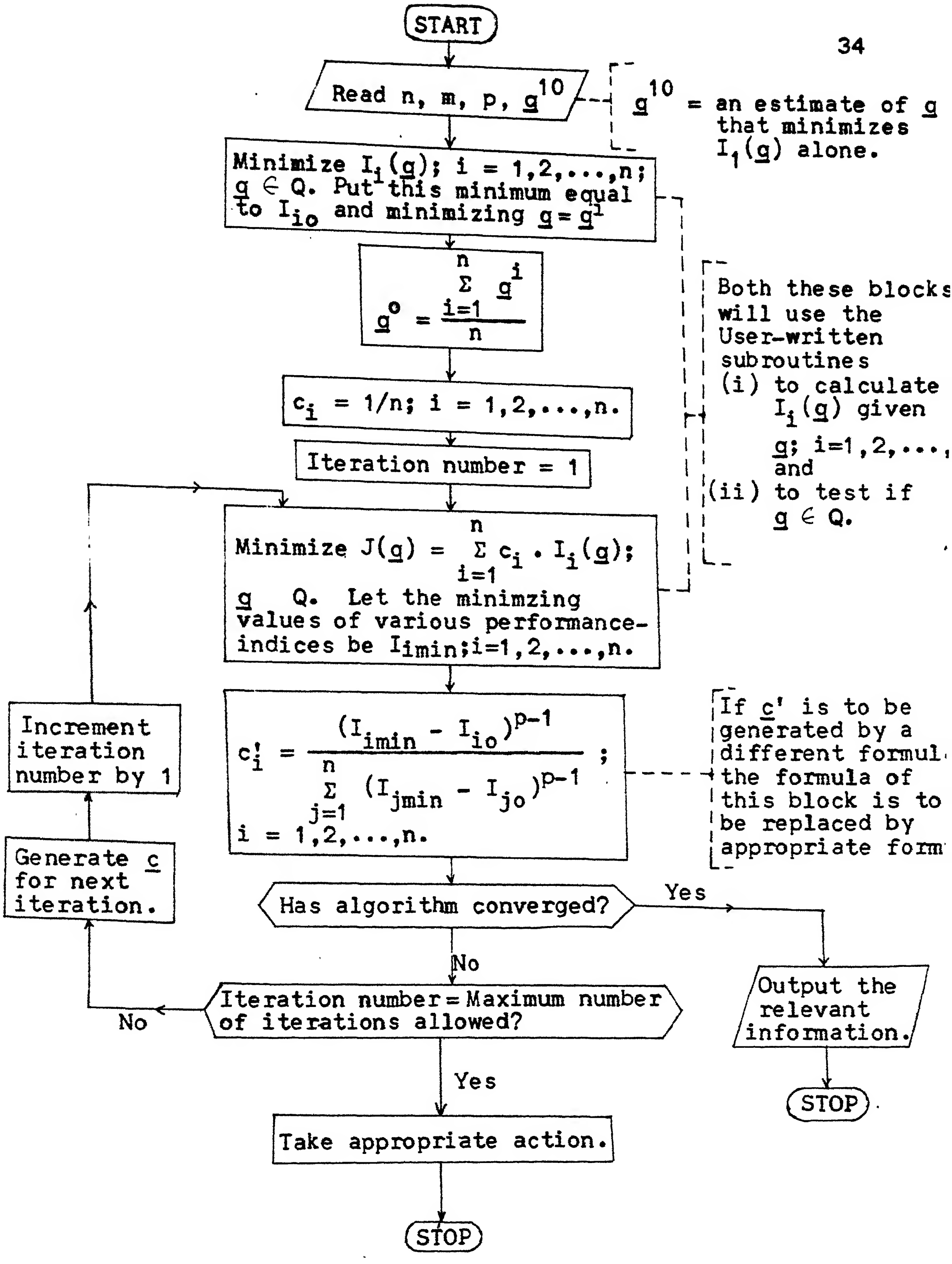
START
Read n, m, p, $\underline{g}^{10}$
$\underline{g}^{10}$ = an estimate of $\underline{g}$ that minimizes $I_1(\underline{g})$ alone.
Minimize $I_i(\underline{g})$; i = 1,2,...,n; $\underline{g} \in Q$. Put this minimum equal to I_{io} and minimizing $\underline{g} = \underline{g}^i$
$\underline{g}^o = \frac{\sum_{i=1}^{n} \underline{g}^i}{n}$
$c_i = 1/n$; i = 1,2,...,n.
Iteration number = 1
Both these blocks will use the User-written subroutines (i) to calculate $I_i(\underline{g})$ given $\underline{g}$; i=1,2,..., and (ii) to test if $\underline{g} \in Q$.
Minimize $J(\underline{g}) = \sum_{i=1}^{n} c_i \cdot I_i(\underline{g})$; $\underline{g}$ Q. Let the minimzing values of various performance-indices be I_{imin}; i=1,2,...,n.
$c_i' = \frac{(I_{imin} - I_{io})^{p-1}}{\sum_{j=1}^{n} (I_{jmin} - I_{jo})^{p-1}}$; i = 1,2,...,n.
If $\underline{c}'$ is to be generated by a different formul the formula of this block is to be replaced by appropriate form
Increment iteration number by 1
Generate $\underline{c}$ for next iteration.
Has algorithm converged?
Yes
Output the relevant information.
STOP
No
Iteration number = Maximum number of iterations allowed?
No
Yes
Take appropriate action.
STOP

Fig. 3.2: Modified Nandakumar's algorithm.

3.4.1 Example 1

This was considered by Nandakumar [5], pp. 76–86. System considered had the transfer function

$$\frac{Y(s)}{U(s)} = \frac{1}{s^2 + 2\delta s + 1} \; ; \quad y(o) = 1, \quad \dot{y}(o) = 0, \; u(t) = 0.$$

Two-dimensional vector performance-index considered was

$$\underline{I} = \begin{bmatrix} I_1 \\ \\ I_2 \end{bmatrix} = \begin{bmatrix} \int_o^\infty e^2(t)\, dt \\ \\ \int_o^\infty (0.25\, e^2(t) + u^2(t))\, dt \end{bmatrix},$$

where $e(t) = y(t) - u(t)$.

Since $\underline{I}$ is to be optimized with respect to δ, it was expressed as follows:

$$\underline{I}(\delta) = \begin{bmatrix} I_1(\delta) \\ I_2(\delta) \end{bmatrix} = \begin{bmatrix} \delta + \frac{1}{4\delta} \\ 0.25\delta + \frac{0.3125}{\delta} \end{bmatrix}$$

Optimal values of the individual performance-indices were calculated analytically by Nandakumar as follows:

$$I_{10} = 1.000, \qquad \delta = 0.5$$

$$I_{20} = 0.559, \qquad \delta = 1.118$$

This information was fed to the program developed in this thesis with $\delta \in [0.4, 1.3]$. The results are shown in

Table 3.1. Since the case of $p = 1$ is trivial, this was not calculated.

Table 3.1

Compromise Solutions for Example 1 Calculated by the Program Developed in This Thesis

$$\text{Utopian Point, } \underline{I}^{o} = \begin{bmatrix} 1.000 \\ 0.559 \end{bmatrix}$$

p	c_1	c_2	I_1	I_2	δ
2	0.451	0.549	1.055	0.623	0.695
3	0.446	0.554	1.056	0.622	0.698
4	0.442	0.557	1.057	0.621	0.700
5	0.438	0.562	1.058	0.621	0.702

Results computed by the program developed (Table 3.1) compare well with the results obtained by Nandakumar (Table 3.2).

Most of the differences between the results of Table 3.1 and Table 3.2 can be attributed to the fact that while Nandakumar minimized $J(\delta) = c_1 I_1(\delta) + c_2 I_2(\delta)$ analytically, the program of this thesis does so numerically.

Figure 3.3 is a plot of I_1 versus I_2. It can be seen that the compromise solutions exist only on the lower boundary of the admissible region $\emptyset$ of the performance-index space, i.e., between points A and B.

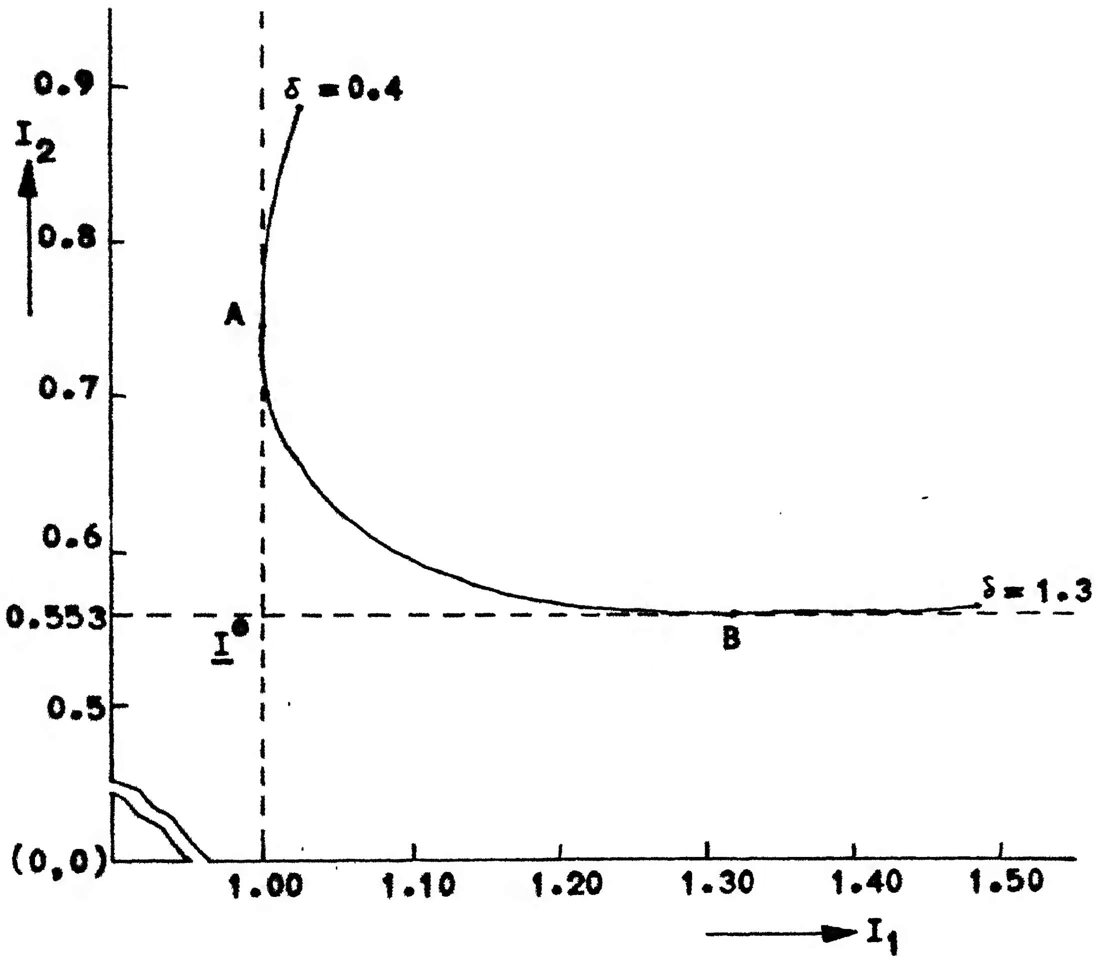

Fig. 3.3: I_1 versus I_2 curve for Example 1.

Table 3.2

Compromise Solutions for Example 1 Computed by Nandakumar [5]

$$\text{Utopian Point, } \underline{I}^{o} = \begin{bmatrix} 1.000 \\ 0.559 \end{bmatrix}$$

p	c_1	c_2	I_1	I_2	δ
2	0.454	0.546	1.054	0.624	0.694
3	0.445	0.555	1.056	0.622	0.699
4	0.441	0.559	1.057	0.621	0.700
5	0.439	0.562	1.058	0.621	0.702

3.4.2 Example 2

The system considered is the second order system given by its transfer function

$$\frac{Y(s)}{U(s)} = \frac{1}{s^2 + 2\delta s + 1}$$

with zero initial conditions and a unit-step input applied at time t = 0.

This system will be considered because its behaviour is well understood.

The two-dimensional performance-index to be considered is

$$\underline{I}(\delta) = \begin{bmatrix} I_1(\delta) \\ \\ I_2(\delta) \end{bmatrix} = \begin{bmatrix} \int_0^{\infty} e^2(t)\, dt \\ \\ \int_0^{\infty} t \,.\, |e(t)| \,.\, dt \end{bmatrix}$$

where $e(t) = y(t) - u(t)$.

The two criteria are not mutually conflicting, except between A and B, i.e., on the lower boundary, as shown in Figure 3.4.

The results using the program developed in this thesis are shown in Table 3.3; in this example, all minimizations including individual minimizations I_{io} were done numerically on computer.

Table 3.3

Compromise Solutions for Example 2 Computed by the Program Developed in This Thesis

$$\text{Utopian Point, } \underline{I}^{o} = \begin{bmatrix} 1.000 \\ 1.952 \end{bmatrix}$$

p	c_1	c_2	I_1	I_2	δ
2	0.700	0.300	1.062	1.977	0.709
3	0.748	0.252	1.058	1.985	0.702
4	0.766	0.234	1.054	1.999	0.693
5	0.766	0.234	1.053	1.999	0.693

Graham and Lathrop [7] tried to show that

$$I_2(\delta) = \int_0^{\infty} t \cdot |e(t)| \cdot dt$$

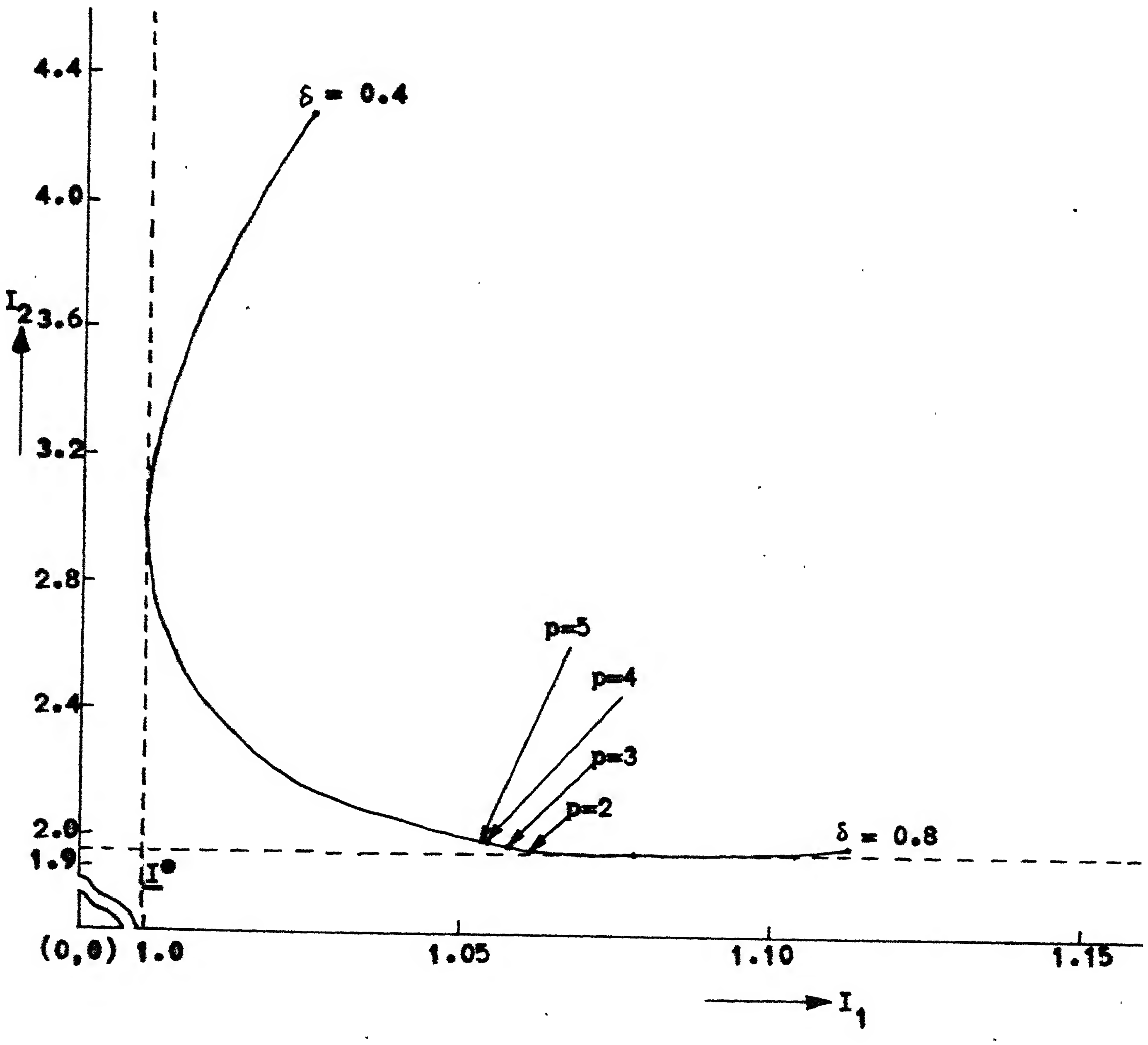

Fig. 3.4: I_1 versus I_2 curve for Example 2.

is superior to

$$I_1(\delta) = \int_0^\infty e^2(t) \, . \, dt$$

in so far as the transient response is concerned. Since compromise solutions lie between A and B in Figure 3.4, a more flexible choice of parameter δ is obtained with 2-dimensional performance-index as compared to using only one index; this point was noted by Nandakumar [5].

CHAPTER 4

CONCLUSIONS

The various properties of Nandakumar's algorithm necessary to develop a computer program have been studied with particular attention to the convergence property. In view of the limited time available, the program developed is applicable only to a relatively small subclass of problems to which Nandakumar's algorithm is applicable.

The sequence proposed to ensure convergence of Nandakumar's algorithm has been successfully tested.

Further work on the subject can be done by developing a very general computer program that can handle all types of problems to which Nandakumar's algorithm is applicable with appropriate additions.

There is also scope for further work on parameter optimization on the lines of Graham and Lathrop [7]. In [7], for single-input-single-output linear systems with zero initial conditions, standard forms of transfer functions (i.e., standard values of the parameters of the system) have to be developed to minimize $\int_0^\infty t \cdot |e(t)| \cdot dt$. It is possible to identify the various performance-indices of importance in moder multi-input-multi-output systems and form vector performance-indices using various combinations of these scalar indices to develop standard designs.

APPENDIX A

VECTOR-VALUED PERFORMANCE-INDEX: A QUALITATIVE REVIEW

A.1 Conventional Approach

To start with, a very brief review of the familiar topic of 'Optimal Systems' follows.

G.W. Leibniz coined the word "optimum" during a philosophical discussion to conclude that our universe is "the best (optimum)" among all possible universes (see Footnote 6). Today we, the control engineers, use the word "optimum" in very much the same sense of Leibniz, albeit in a general way by using it in both "the best" and "the worst" sense, depending upon the problem.

Of course, man's desire for perfection ("optimization") is much older than the time of Leibniz. To cite but only one example, many ". . . ancient cities are in fact circular, probably to minimize the lengths of city walls needed to enclose the city's fixed area" [8 , p. 3]. But most such ancient optimizations are based on intuition which will most often fail in complex problems. So we, the control engineers, are more

---

[6] Leibniz invented the word "optimum" in his book (published in 1710) "Theodicy: Essays on the Goodness of God, the Freedom of Man, and the Origin of Evil" (translated by E.M. Huggard, and translated edition published by Yale University Press, New Haven, 1952 (p. 128)), according to [8 , pp. 4 and 9].

concerned with quantitative analysis (instead of intuitive approach) which is relatively easy to use and is more reliable in extremely complex systems.

For a quantitative study of optimal control systems, three well-recognized steps are usually employed by modern optimal control system theorists:

1. Getting the information about the system, accurately and quantitatively, as to how the system variables interact. This often involves developing a suitable, often much too simplified, mathematical model of the system involving system variables.
2. Deciding on how to measure system performance quantitatively in terms of system variables. This almost always means deciding a single number, called performance-index, the value of which is accepted as the quantitative measure of system performance. Performance-index should, of course, be a function of those system variables (may be functions of time) the values of which can be altered to get better performance out of the system.
3. Changing the relevant system variables such that the performance-index is optimized.

In each of the three steps just described, the constraints on system variables (which are almost always imposed in some sense) should be borne in mind.

A.2 More On Performance-Indices

Three steps in optimization problem have just been described. Step 1 requires considerable effort for complex systems and is certainly a difficult problem; techniques that can reduce this effort or increase its accuracy are most welcome, but this is not the problem of interest in this thesis. Many efficient algorithms to implement step 3 (optimization, i.e., minimization or maximization) are known when performance-index is a single number, as is the case just considered. For the time being, however, only step 2, finding a measure of system effectiveness, shall be addressed.

"Optimization is decisive because it narrows down the possible choices to one - the best one" [8 , p. 3]. How heavy the dependence of the decisiveness of optimization is on the performance-index chosen cannot be overemphasized. So more realistic a performance-index for a given system, more accurate and realistic the results.

So a considerable research effort has been spent on what a performance-index should be for a specific system (or more often, for a class of systems), since the use of performance-index was introduced in control systems by Norbert Wiener during World War II. So, from the simplest performance-indices chosen earlier for single-input-single-output systems, refinements were made (and are still being made) on the performance-index, many times making it more complicated and more realistic but with more requirements on computational effort and new techniques for its computation.

Design of a specific performance-index, in general, requires

(i) identification of desired system characteristics.

(ii) finding a numerical measure to measure each of these characteristics. This numerical measure should be calculable from the numerical values of those system variables with respect to which the system is to be optimized.

(iii) giving appropriate weightage to different characteristics of the system to find out a single number obtainable from numerical measures of various system characteristics. This final number, the performance-index, is therefore calculable from those system variables subject to which the optimization is to be carried out.

Stability (and how stable a system is), sensitivity, accuracy, transient response, resonance characteristics, bandwidth, size, weight, power consumption and economics are amongst the numerous characteristics desired from a control system [9 , pp. 75-77, 79, 89, 101-102; 10 , p. 3]. However, the importance accorded to specific characteristics varies from system to system [9 , pp. 75, 79], the overall goal in most physical systems being profit, cost or efficiency [8 , p. 3].

Designing a suitable performance-index means measuring quantitatively the various characteristics and giving them appropriate weightage; in many systems, this may be extremely difficult [8 , p. 3]. So, most of the older literature on this subject considers a single-input-single-output system with performance-index formed by functionals (of varying complexity) of system error with the conditions that the system be stable and physically realizable. A review of many important criteria of this type can be found in [7 ; 9 , Chapter 4; 10 , Chapter 1];it can be seen from this literature that as the performance-index is made more and more realistic, it gets more and more complicated. Many performance-indices have been suggested during recent years for modern complex multi-input-multi-output systems to make them more and more accurate. But complicacy of the practical systems renders the job difficult.

A.3 Need for Vector-Valued Performance-Index

A few problems of scalar performance-index have been noted in Section A.2. There are more problems with a scalar performance-index; let us note the views of a few authorities on the subject.

1. "One of the most serious weaknesses of the current theories of optimal control is that they are predicated on the assumption that the performance of the system can be measured with a single number ....

   The trouble with this concept of optimality is that, in general, there is more than one consideration that enters into the assessment of performance of ..." the system "... and in most cases, these considerations cannot be subsumed under a single scalar-valued criterion. In such cases, the system S may be superior to the system S' in some respects and inferior to S' in others ...." [1 , p. 59].

2. "Classical control theory, which uses transform methods to solve systems of linear equations with constant coefficients, handles multicriteria problems in an implicit manner. The various performance-indices to be considered in a classical problem are merely evaluated directly at each step of the trial-and-error design of a

suitable control system. Modern or optimal control theory is superior to the classical method because the optimal control method is capable of handling more general types of nonlinear and linear control systems. Optimal control theory also provides a means of optimizing a performance criterion whereas the classical theory offers only a trial-and-error scheme for selecting a linear system to satisfy the predetermined performance restrictions. However, the use of scalar performance-index in the formulation of the optimal control problem has tended to obscure the traditionally multidimensional character of the performance measure." [2 , p. 26].

3. "... disadvantage in ..." optimal system design with a scalar performance-index "... approach comes from the fact that performance of the system must be represented mathematically by a single scalar-valued performance-index. Since there are usually many requirements that enter into the judgement of system performance, the choice of a single index to represent one or more of these requirements is necessarily arbitrary or subjective. Unless the trade-offs achievable between various competing requirements are well understood, there is little objective basis for specifying the performance-index ...." [11 , p. 469].

4. "... control rules that are well defined through conditions of optimizing a single criterion leads to the following: The selection and optimization of a given functional answers to only one of the requirements of the control system, while other requirements, which are often equally important are ignored. The totality of all requirements may be accounted for by a collection of functionals forming a vector of criterion functions, i.e., a vector-valued criterion." [3 , p. IX, Introduction].

5. "Even when the type of ..." system performance "... measure is clear, it is not always easy to express its quantitative dependence on system variables." [8 , p. 3].
In such cases, it may be easier to express quantitative performance measure as a vector instead of scalar.

All these quotations underline the importance of using a vector-valued performance-index in optimal control theory instead of a scalar performance-index.

A.4 Peculiarities of Vector-Valued Performance-Indices

Discussion of this and the next section will be with respect to the problem defined in Section 1.2, Chapter 1.

Following notes pinpoint the basic differences between scalar and vector performance-index optimization. They also offer a general outline of the characteristics of vector performance-indices.

A.4.1 Note 1

"In the case of scalar performance-index, one is interested in a point on the real number line that represents the optimum index value, and ..." $\underline{v}$ "... that corresponds to this index. The extension of this investigation to the case of a vector performance-index naturally leads to the determination of an optimum index surface and the corresponding ..." $\underline{v}$ in an n-dimensional performance-index space. "The index surface to be determined represents the end points of index vectors from which the control designer can make his final selection." [2, p. 1

These statements stem from the fact that no two vector performance-indices can be compared the way scalar performance-indices are compared to give a unique optimum solution. To define the above-mentioned surface of "optimum" vector performance-indices, a "optimum" vector performance-index will have to be defined; so the definitions of Note 2.

A.4.2 Note 2

When working with vector performance-indices, the difference is usually made between "noninferior" and "optimal" systems or performance-indices. In the context of this thesis, a performanc index $\underline{I}(\underline{v})$ will be considered "optimum" if all of its components are minimized simultaneously (by a single $\underline{v}$). And a performance-index $\underline{I}(\underline{v})$ will be considered "noninferior" if there is no performance-index whose every component is less than the corresponding component of $\underline{I}(\underline{v})$.

A.4.3 Note 3

It is clear from Note 2 that the "optimum index surface" of Note 1 is the set of non-inferior systems. In this context, following quotation is of interest:

"Usually, the designer will have enough physical intuition concerning the problem to decide whether or not an optimal solution exists. If, for example, he considers a simple rocket problem, he can intuitively recognize fuel consumption and kinetic energy as comparable performance-indices. It is when criteria are found to be conflicting, either by intuitive reasoning or by trial-and-error, that the concept of a non-inferior set becomes important" [2, pp. 15-16].

The curious point is: if various performance-indices are compatible, it is as good to minimize one as to find the optimum vector. It is only when one is searching for a system subject to many conflicting criteria (which is most often the case) that vector optimization theory has the greatest role to play.

A.4.4 Note 4

Whether one is working with a scalar performance-index or a vector performance-index, the ultimate goal is a single solution as the system to be implemented. So, in vector optimization, a single point for ultimate design from the noninferior "index surface" of Note 1 is to be selected.

Going through literature, it is found that virtually every author suggests his own criterion. For example, Zadeh [1] suggests that a scalar performance-index should be introduced over the set of non-inferior systems to make the ultimate choice. Reid and Citron [2] offer a more original approach: "subjectively" select the desired system from the set of non-inferior systems because when "... subjective criteria are concerned (e.g., safety versus profit) ..." this "... method of attack seems more plausible, because it leaves the subjective questions to be answered by subjective methods ...." It appears that although we should strive for different algorithms or criteria to select a single system out of the set of non-inferior systems, ultimate choice of how a system will be selected should be left to the designer; a universally applicable standard method appears a difficult-to-realize dream.

A.5 Present Information About the Subject

Although the "... first formulation of a problem of optimizing vector-valued criteria was given in 1896 ..." by "... economist Pareto ...", it was only in 1963 when Zadeh published his article [1] "... which first presented the question of designing control systems which are optimal relative to several performance-indices." [3, p. 1]. So, the study of optimal control systems with vector criteria is relatively new.

The "... goal of L.A. Zadeh's paper, in addition to its initiatives, is to show that in a majority of practical cases, <u>exact optimization of a vector functional is unattainable</u> ...." This means that if a $\underline{v}$ "... is chosen to optimize one scalar functional, then almost always it is not possible to optimize a second scalar functional using the same ..." $\underline{v}$. This means that "... the problem of optimizing all scalar criteria with the same ..." $\underline{v}$ "... is impossible and to give it a precise mathematical formulation is also impossible ...." [3, p. 11].

At present, the question of existence of noninferior solutions under vector-valued performance-index is well settled [3, pp. 2, 20-23, Chapter 3; 5, p. 2].

Following is a brief qualitative survey of the various approaches being used in solving vector optimization problems; a much more detailed survey can be found in [3, Chapter 1].

Various methods proposed for the solution of vector-valued performance-index optimization can be classified into the following groups [5, p. 2; 3, p. 3]:

1. Optimization of an hierarchical sequence of performance-indices is based upon the idea "... of a preference ordering of the given criteria ...." Although a great improvement upon conventional single scalar-valued performance-index optimization techniques, this method does not realize the full potential of vector-valued performance-index optimization "... since optimization with respect to the first, most important criterion already leads to a unique optimal solution and everything is reduced to optimization only relative to the first criterion ...." [3, p. 4]. This thesis does not use this method.

2. The method of determination of a set of noninferior points is the most vastly explored one, the idea of which was first given by Zadeh [1].
   A landmark in this technique is the paper by Reid and Citron [2], discussed in Section 1.3, Chapter 1.
3. The methods for determining a solution based on some form of compromise are in existence for some time. Many forms of compromises have been proposed. For example, Borisov's principle of equitable compromise is "... based upon the idea that the relative decrease in the value of one or several criteria should not exceed the relative increase in the remaining criteria." [3, p. 7].
   Another compromise approach is the "method of constraints" in which "... one starts with a solution relative to which it is necessary to impose constraints on particular criteria which must be satisfied in the first place. On the basis of this information, a revised solution is affected." [3, p. 9].
   One of the most important ideas of compromise solutions is due to Salukvadze [3, 4] discussed in Section 1.4, Chapter 1.

APPENDIX B

PATTERN SEARCH ALGORITHM TO MINIMIZE A FUNCTION OF n VARIABLES

Reference: [8, pp. 307-310]

This is a direct climbing type of technique. Algorithm "... is based on the hopeful conjecture that any ... adjustment of independent variables, which have been successful during early experiments, will be worth trying again."

"Although the method starts cautiously with short excursions from the starting point, the steps grow with repeated success. Subsequent failure indicates that shorter steps are in order, and if a change in direction is required, the technique will start over again with a new pattern. In the vicinity of the ..." bottom "... the steps become very small to avoid overlooking any promising direction."

Algorithm is as follows.

Suppose we want to minimize $y(\underline{x})$, where $\underline{x}$ is n-dimensional. Illustrations of the method will be for 2-dimensional $\underline{x}$ (Figure B1).

Searth of $j^{\underline{th}}$ iteration is started at a point (called "temporary head") $\underline{t}_{jo}$, with a step-size δ_i for $i^{\underline{th}}$ variable. Let $\underline{\Delta}_i$ be the vector whose $i^{\underline{th}}$ component is δ_i and the rest of components are zero. We perturb the $i^{\underline{th}}$ variable x_i about the point $\underline{t}_{j,i-1}$ (starting from x_1) and find a new temporary head $\underline{t}_{ji}$ such that

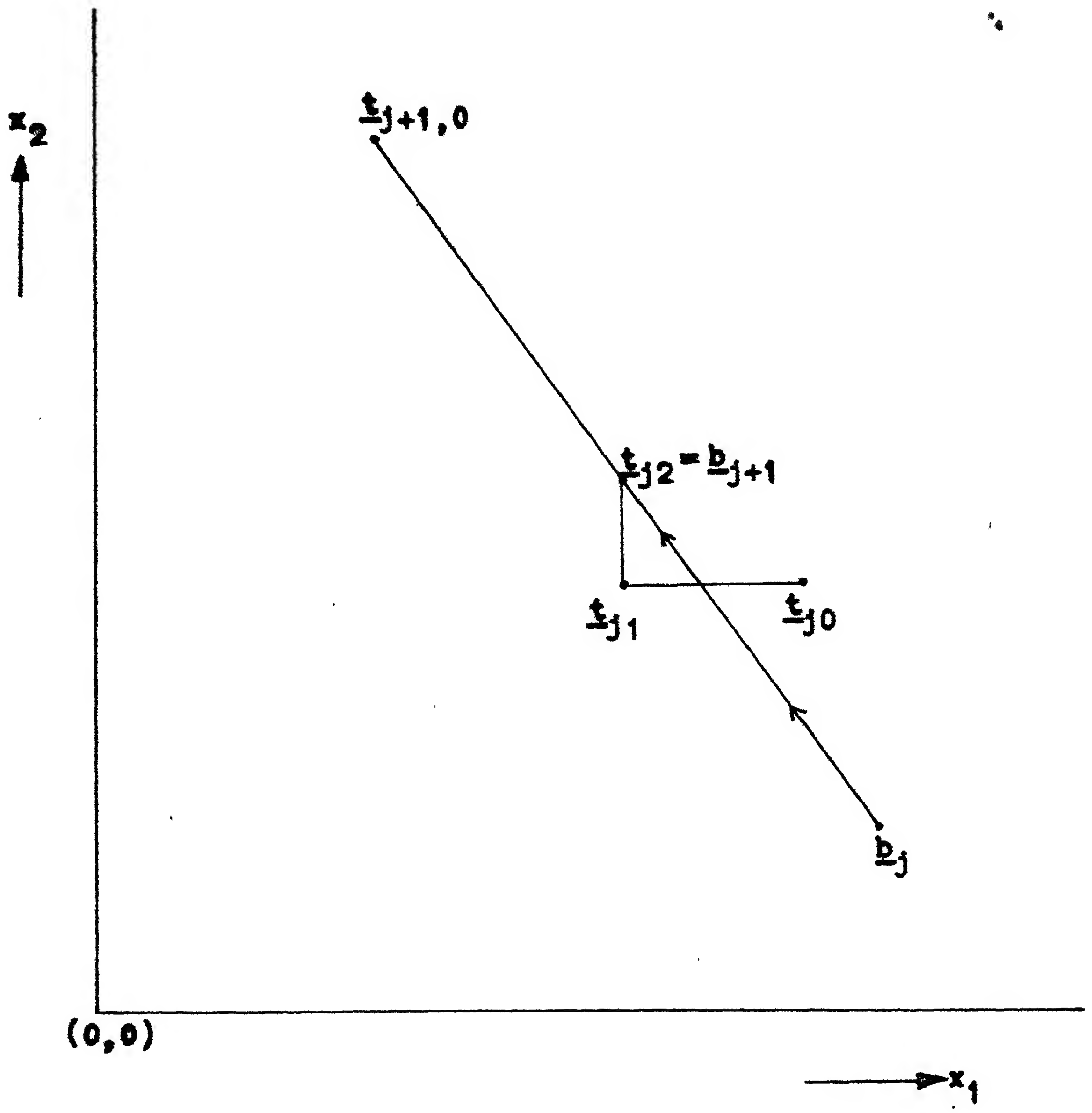

Fig. B1: Illustration of an iteration of Pattern Search Algorithm.

$$\underline{t}_{ji} = \begin{cases} \underline{t}_{j,i-1} + \underline{\Delta}_i, & \text{if } y(\underline{t}_{j,i-1} + \underline{\Delta}_i) \leq y(\underline{t}_{j,i-1}) \\ \underline{t}_{j,i-1} - \underline{\Delta}_i, & \text{if } y(\underline{t}_{j,i-1} - \underline{\Delta}_i) \leq y(\underline{t}_{j,i-1}) \\ \underline{t}_{j,i-1}, & \text{otherwise.} \end{cases}$$

When all variables have been perturbed, the "base point" for next iteration is defined by

$$\underline{b}_{j+1} \triangleq \underline{t}_{j,n}$$

Let the base point for j<u>th</u> iteration was $\underline{b}_j$. Starting temporary head $\underline{t}_{j+1,0}$ for next iteration is found by joining $\underline{b}_j$ and $\underline{b}_{j+1}$ by a straight line and doubling it in the same direction, i.e. by

$$\underline{t}_{j+1,0} \triangleq \underline{b}_j + 2 \cdot (\underline{b}_{j+1} - \underline{b}_j)$$

If $y(\underline{t}_{j+1,0}) \leq y(\underline{b}_{j+1})$, (j+1)<u>th</u> iteration is started as before. If it is not so, $\underline{b}_{j+1}$ is obviously the minimum point till now; and no similar search pattern can be continued. The idea then is to start anew from point $\underline{b}_{j+1}$ as new base point $\underline{b}_{j+2}$ and starting temporary head $\underline{t}_{j+2,0} = \underline{b}_{j+2}$, but with shorter steps. The process is continued till the step-size becomes so small that required accuracy is assured.

The algorithm is started with an initial base point $\underline{b}_1$ (which may be arbitrarily chosen), with $\underline{t}_{10} = \underline{b}_1$.

REFERENCES

1. L.A. Zadeh, "Optimality and non-scalar-valued performance criteria", IEEE Trans. Automat. Contr., Vol. AC-8, pp. 59-60, Jan. 1963.

2. R.W. Reid and C.J. Citron, "On noninferior performance index vectors", J. of Optim. Theory and Appl., Vol. 7, pp. 11-28, Jan. 1971.

3. M.E. Salukvadze, 'Vector Valued Optimization Problems in Control Theory' (Book), Academic Press, Inc., New York, NY, U.S.A. (1979).

4. M.E. Salukvadze, "Optimization of vector functionals. (I) The programming of optimal trajectories", Automat. Telemekh. (English translation entitled 'Automation and Remote Control' is available.), Vol. 32, Pt. 1, Aug. 1971.

5. M.P. Nandakumar, "Optimal control systems design with vector-valued performance-index", Ph.D. Thesis, Deptt. of Elect. Engg., I.I.T., Kanpur, Sept. 1986 (Degree on this thesis is yet to be awarded).

6. E. Kreyszig, 'Advanced Engineering Mathematics' (Book), Fifth Edition, Wiley Eastern Ltd., New Delhi (1985).

7. D. Graham and R.C. Lathrop, "The synthesis of 'optimum' transient response: Criteria and standard forms", Trans. AIEE, Vol. 72, Pt. II (Applications and Industry), pp. 273-288, Nov. 1953.

8. D.J. Wilde and C.S. Beightler, 'Foundations of Optimization' (Book), Prentice-Hall, Inc., Englewood Cliffs, N.J., U.S.A. (1967).

9. S.M. Shinners, 'Control System Design' (Book), John Wiley and Sons, Inc., New York (1964).

10. B. Sarkar, "Numerical and algebraic methods for computer-aided design of linear and piece-wise linear systems", Ph.D. Thesis, Deptt. of Elect. Engg., University of British Columbia, May 1967.

11. W.L. Nelson, "On the use of optimization theory for practical control system design", IEEE Trans. Automat. Contr., Vol. AC-9, pp. 469-477, Oct. 1964.

EE-1987-M-DHO-SOM